FUNDAMENTALS OF OBJECT ORIENTED PROGRAMMING

Lecture58

Lecture 58: Case Study - Online Shopping Cart Java

Bye. So, welcome to lecture number 58. So, we are studying case studies in object-oriented programming. In the last two lectures, we saw extensive applications in C++ and And in today's lecture, we will talk about the online shopping cart system using Java.

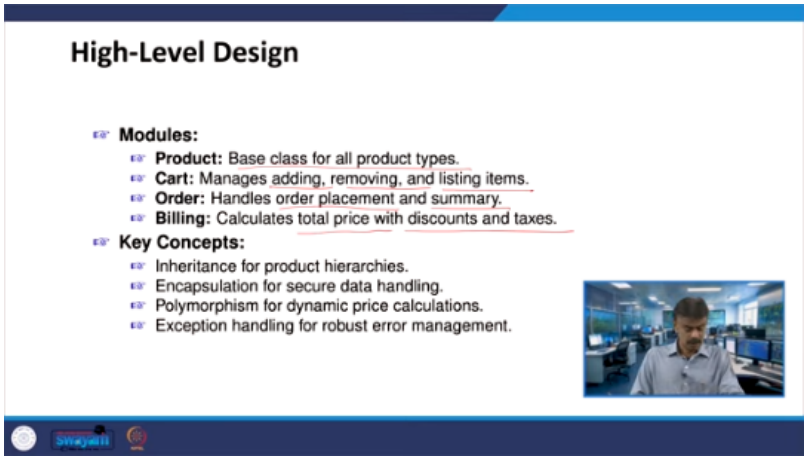
Right. So, the main objective is to build an online shopping cart system using object-oriented programming in Java. Right. So, the key features we will use are, let us say, product hierarchies. Right.

It may be electronics. It may be clothing. Right. It may be, let us say, some kitchenware items, etc. Right. And then you have cart management, which will add or remove items. Right? So, when you go to the shop, you may pick or remove items. Right? You may not require them, or let us say, you pick 10 items and put one or two back. Right? So, the same applies here, and that is called cart management. The third one is dynamic pricing. Right? The dynamic pricing calculation is exactly what we did in the

Hotel Management System, so this is based on discounts and taxes. All right, so then here also we are going to have the file handling system for saving the orders. Use cases: we are going to simulate the e-commerce functionalities, and we are going to provide a modular design for extensibility. So here, the high-level design, we have the modules. Right. So, the product is a base class for all. Right. We are going to have the modules: Product, which is a base class for all product types.

Right. And then Cart, which is managing adding, removing, and listing the items. You call it a Cart module. And then Order module, this is handling order placement and summary. And then Billing, so this is calculating the total price with the discounts and taxes, right.

So, these are all the four modules: Product, Cart, Order, and Billing. So, the key concepts here: we use inheritance, right, for product hierarchies. And encapsulation for secure data handling, right, encapsulation. So, we use inheritance first for product hierarchies. Encapsulation for secure data handling, then polymorphism for the dynamic billing, right?



So, dynamic price calculations and then exception handling for error management, right? So, we are going to study or we are going to use inheritance, encapsulation, polymorphism, and exception handling. So, this we are going to see with the help of Java. So, what are all the classes we can make? Right.

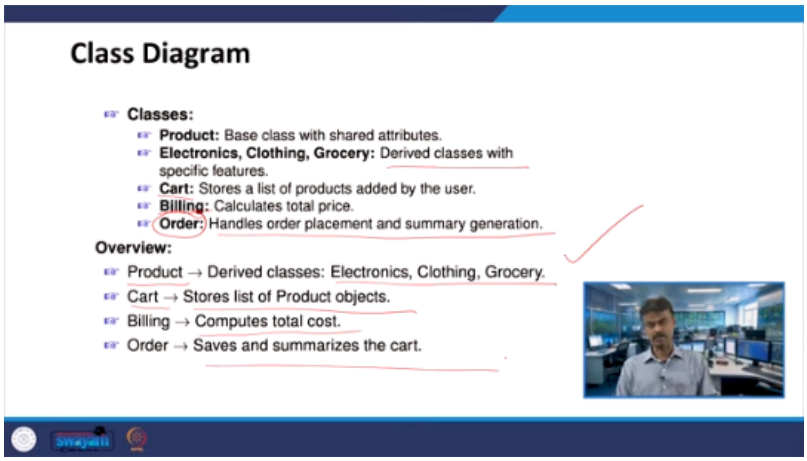
Product. So, product will be the base class. Right. Product is a base class with the shared attributes. Then, we have electronics, clothing, grocery, etc.

They will be the derived classes with the special features. Another class called cart. Right. Which is storing a list of products. Right.

So which is chosen by the user or which are all added by the user. Then billing, right? So, once you have selected the complete items, the price will be calculated, right? So, then you are going to have the billing, right? So, next one is this will handle order placement and summary generation that is you are going to have the order class, right?

So, this will handle order placement and then summary generation that means product is the base class. And the derived classes are electronics, clothing, and grocery. Cart, which is another class, which is storing list of products. And billing, which is computing the total cost. And order, saves and summarizing the cart.

So whatever the user has chosen in the cart, so that will be saved and summarized. So these are all the classes that we are going to make. And then we will go to the main program. and we will run the code. So, let us have product class which is the abstract class product right public abstract class product right.



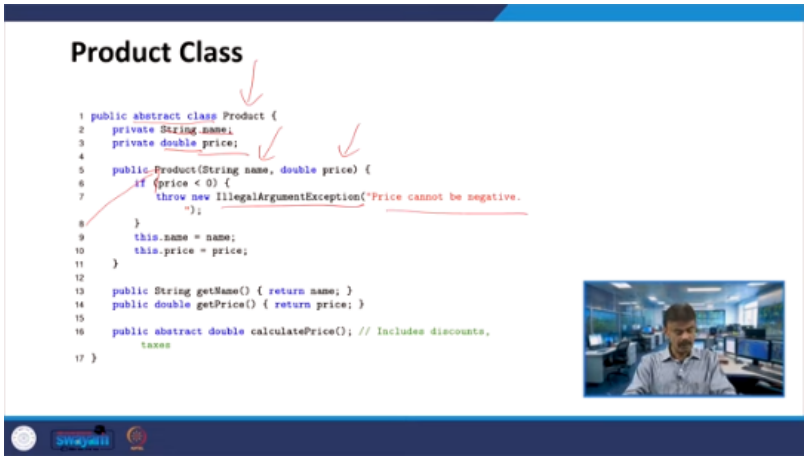
So, we have studied about abstract class if you recall, right? So, 0 to 100 percent abstraction is possible, which means we can also write the regular method, right? Whereas, an interface is 100 percent abstraction. So, here we are using the abstract class product. So, under the abstract class product, you have name, whose data type is string.

Price, whose data type is double, right? So here you have the constructor product, right? So, string name and double price, right? So, name is string and price is double. If price is less than 0, right?

If price is less than 0, so here you go, right? Yes, price less than 0 is not possible, right? Or equal to 0 is also not possible. So, in that case, Yeah, sometimes you may get free.

Yeah, that is possible. Equal to 0 is possible. So, less than 0 is not possible. Right. So, in that case, it will throw this exception: illegal argument exception.

Right. By printing, price cannot be negative. It is understood. Otherwise, name will be copied into this.name, and price will be copied into this.price. Okay.



This pointer that we are using, and this is happening. Then, let us say get name, function get name. Whose return type is string, that will return name, right? And then here you have get price, whose data type is double, and it is returning price, right? So, here you have one abstract function, right?

Public abstract. This is a regular function, right? So, in fact, this is constructor, two argument constructor, correct? And here you go the abstract function, right? So, that means, in fact, abstract method.

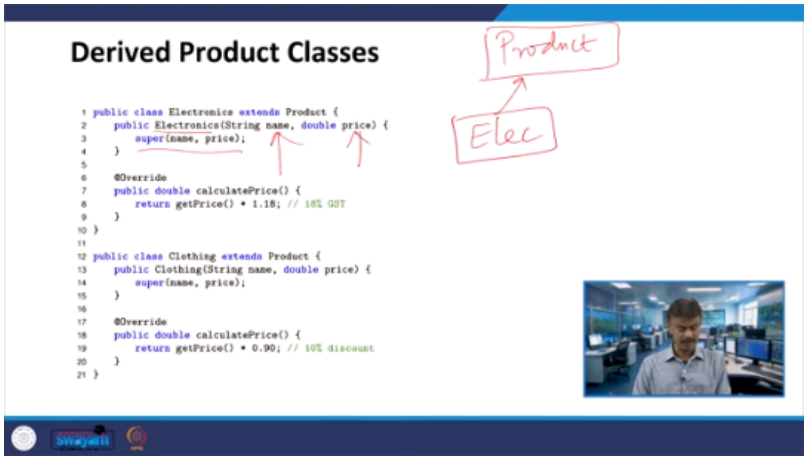
It is a Java. It is Java. Abstract method. right. So, in the previous example, we use virtual function, if you recall, right.

So, here it is abstract method, calculate price, that means, it has to inherit further, right. In the further inheritance, this has to be defined, I mean to say, correct. Anyway, it will be inherited because it is abstract class, I cannot create any object. So, now, electronics, so product is a base class, right. Product is a super class, product is a super class.

And from here, I have All right. So here you have electronics. So this is inheriting publicly the product. Right.

And then here you have constructor electronics. You have the arguments name and price. Right. So super of name comma price. Right.

right. In fact, in the super class constructor product, what is the constructor product? So, this will be called, right. So, name will be assigned to this dot name and price will be assigned to this dot price, right. So, in case you are having the object under electronics, correct.



So, now I have an overriding function calculate price over here. Yes, anyway, this is abstract. So, here when I am writing and assume that you have an object under electronics or you are upcasting under electronics. So, this will be called right—the calculate price, the base, the superclass product right—under this you have method calculate price which will be overridden. Right, that will be overridden, and here the calculated price is being defined whose return type is double, so this will return get price into 1.18, right? 18 percent GST, 1.18, right? So the next one you have clothing. Next class is clothing, which is extending product, so that means this is inheriting product. All right, so I have a clothing, right? This is, in fact, I have to put like this: clothing.

All right. This class. Class is clothing, which is inheriting product. Again, it has the constructor clothing. Name and price you are passing.

All right. So, superclass constructor. Superclass constructor you have name, which will be copied into this dot name. Price, which will be copied into this dot price. All right.

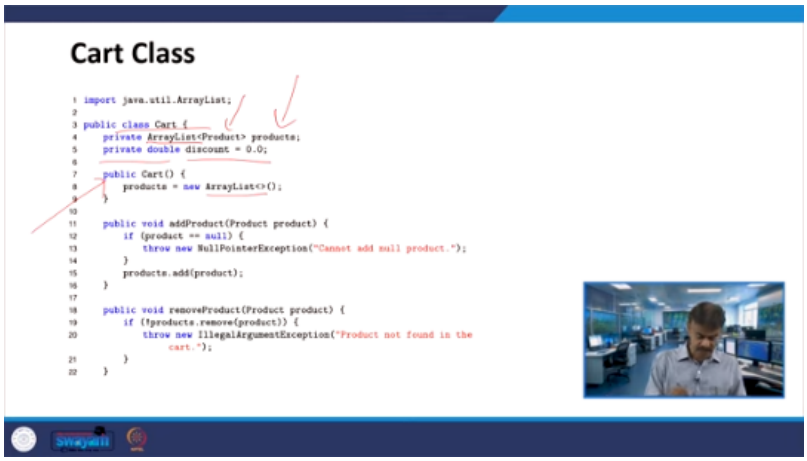
And then here also you have The overriding method, calculate price. So here, if you have a 10% discount, that's dynamic, that's called dynamic pricing. All right. So a similar thing we have done in the hotel management system also.

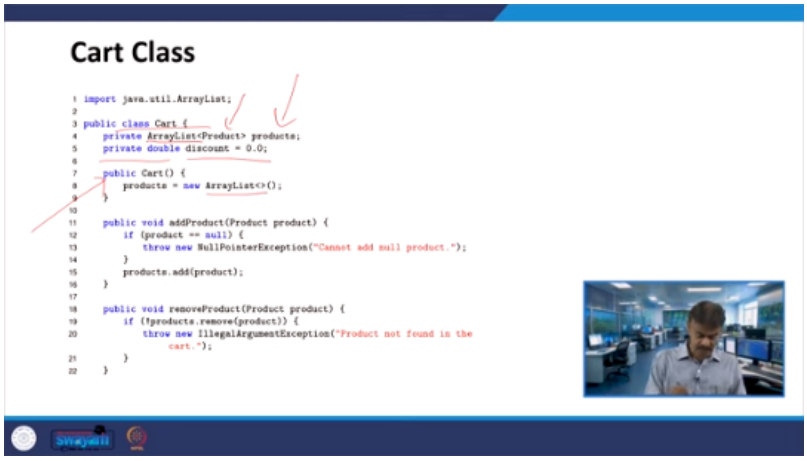
All right. So the get price will be multiplied by 0.90. All right. So get price is over here, which is returning price in line number 14. So, there also line number 8, you have get price, the same meaning get price will be called, it will be returning price.

So, price will be multiplied by 0.9 because here you have a 10 percent discount applied. So, now the next class is a cot class. So, under the cot class. So, here you have public class cot, you have a list of products, that is the array list again, product product is the object small p. And you have a private member data called discount.

Initially, it is 0.0. And here you have the constructor cart, right? So, the products dynamically allocating a memory with ArrayList, right? The ArrayList, right? Which is the constructor, right?

The class is cart. And here you have the ArrayList. So, upcasting to ArrayList products that you are defining in ArrayList. ArrayList of products. Constructor is ArrayList.





So that is your cart constructor. So it is equivalent to this ArrayList product products which is equal to dynamically allocating a memory new and this is a constructor ArrayList. So next you have one method in the class cart which is

called add product. So product is an object. and class is product right if product is equal to null right so product is pointing to null then it is throwing cannot right this is throwing the exception called null pointer exception right which is a inbuilt exception null pointer exception in java right so which is printing cannot add null product you cannot add null product right otherwise you are calling add function

right, add method by pausing the product, add method pausing the product, right, and it is invoking add method by pausing the product. So, the next line, line number 18, you have the method called remove product, right, remove product, you are pausing product. So, if products dot, I mean products is invoking remove product, if this is false, not of this is true, right, if products is invoking remove product, If it is false, not of false is true, then it is throwing the illegal argument exception. So product not found in the cart.

If it is not there, right? So that means the remove product is becoming false. If nothing is there, you can't do remove product, right? So it will become false. So not of false become true.

So then in that case, it is throwing false. this illegal argument exception so that means product not found in the cart will be displayed so this is cart class and we have some more functions right methods get products all right so list of array list of product return products all right and then here you have apply coupon this is the method and string code right so code is the parameter that you are passing whose data type is string if it is equal code dot equal save 10 sometimes right you are going to pass this and it is equal to save 10 so discount is equal to 0.10 right otherwise it is invalid coupon code what is the code right you are writing the code other code there is no coupon discount ok and here you have method called calculate total Total is initialized to 0. Then you have for loop.

Assume that is arranged for a loop. You have so many products. All the products are going into the product. So both references are pointing to the same address. And total will be equal to total plus product dot calculate price.

So for all the prices. So you are adding. Right. So those you are adding. All the product.

So, here you have line number 15. Assume that P1, P2, P3, Pn are there. All the prices will be added. So, then total into 1 minus, suppose discount is 0.1. So, here you go.

Save 10 coupon. So, 1 minus 0.1, that means 0.9. 0.9 will be multiplied by total. That means you are getting 10% discount. That is calculate total.

So, then you are going to have Order class. Right. So we will see order class. So class order.

Your object is caught. Right. Under class caught. Here you have the constructor. right constructor object cart right under cart so cart is copied into this dot cart and then here you have a method called place order right so your printing order summary range for loop so assume that you have all the get products right when the cart is invoking get products that will be copied the product right so you have several products assume that is a p1 p2 p3 etc right so all will be printed that means assume that you have n products the name

Get the name and print the value. All right. So, in the dollar, what is the value? Calculate the price. So, it is invoking calculate price.

Let us say dollar 10, dollar 20, dollar 5, etc. All right. So, then you are printing. So, the total dollar is the cart is invoking calculate total. The cart is invoking calculate total.

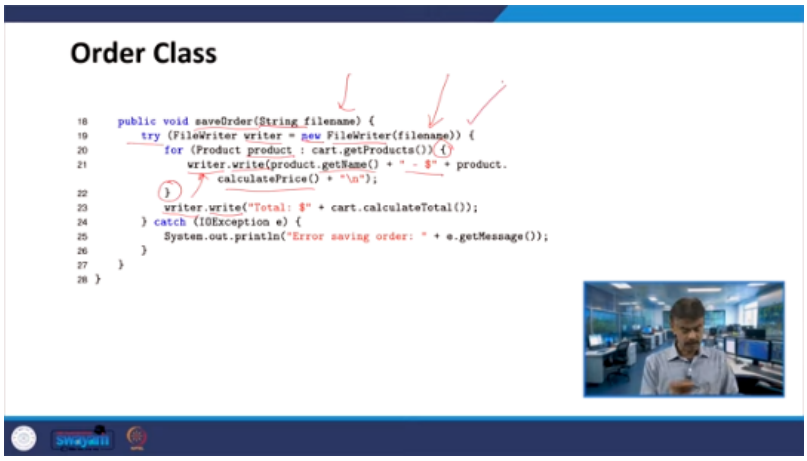
Calculate total, we have done. And then save the order. Exactly what we have done in the case of the hotel management system. That particular problem. We are almost going to do the same thing.

Here you have save order function, save order method, file name under string. So, we are going to handle the file, right. If, I mean, you are trying, you have writer object under file writer, dynamically allocating a memory with the constructor file writer, you are, right, going to pass the file name. So, that we will see. Maybe it will be available in the main.

What is the file name? Some .txt or something. right and then we are again the same the get products so that is copied into product and then writer object which is invoking write method so you will have a get name dollar value on the calculate price right so this is for the range for loop so assume that you can able to do

successfully right so these rank number 20 21 22 will happen right and then the writer object will invoke write right writer object will invoke write. So, this is up to here your try is still continuing right.

So, your try is starting here correct. So, it is continuing line number 23 is under try only. So, it will write right will call write method writer is object writer will call write method total dollar is cart is invoking calculate total right. If there is any problem if you are not able to write open the file or the file does not exist All right.



It will throw the exception. The exception will be caught here, and then the error saving order and e.getMessage(). So whatever getMessage that we had seen previously, that will be printed. These are all, you know. Right.

So now, go to the main program. Right. Main class. So this is a derived class. Here you have the main method.

So I have an object called cart under the class Cart. Dynamically allocating memory, right, with a constructor Cart, right. So, here I am trying System.out.println, this will print adding valid products, right. Now, cart is invoking, this object is invoking addProduct, So here you go, the constructor Electronics, where you are passing laptop and thousand.

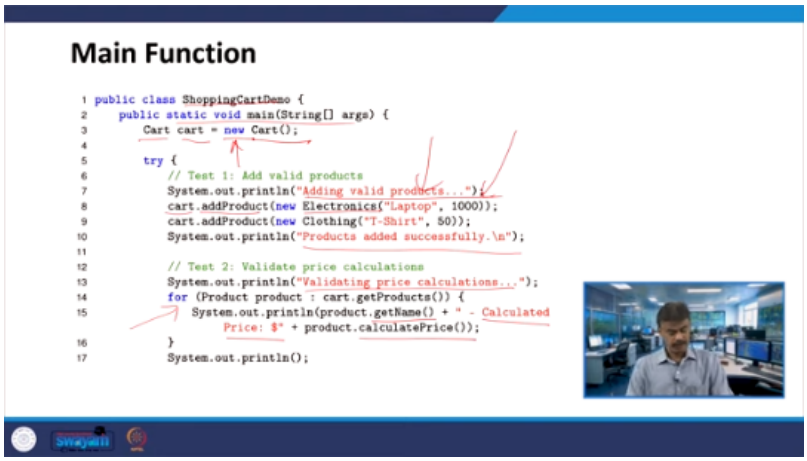
So we have seen this: electronics, clothing, all the classes because I have written, I remember. So here you go, class electronics. And then here you have the constructor, you are passing name and price. So that is what you have done: laptop, thousand. Name is laptop, and then name and price, thousand.

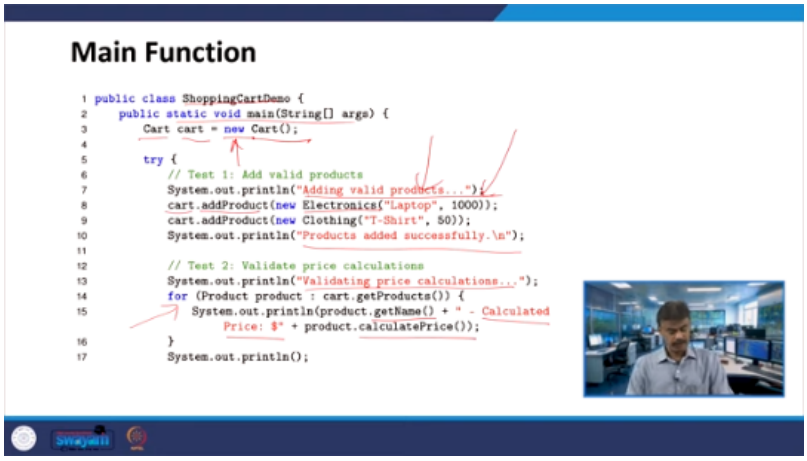
$1,000. All right. Laptop, $1,000. Clothing, T-shirt, $50. Add product.

Right. So then products are added successfully. You are adding the valid product. That's a test one. All right.

So test two, validate price calculation. So you are printing, validating price calculation. Here you have the for loop, the products again. Let us say P1, P2, P3, PN. So the reference will be product.

All right. So the range, all the ranges is a range for loop. All the ranges will be printed. So it is invoking get name, calculated price in dollar, and the product is invoking calculate price. So then you have system.println over here outside.





Your range for loop is here. So one statement, system.out.println, new line, line number 17. And line number 18, you have system.out.println attempting to add a null product. So try it. Attempting to add null product.

And then what will happen? So this particular case it is handling invalid product addition. So that means when you are pausing null value. Add product. When it is having null.

This we have seen. Add product. If product equal to null. Line number 12. So then it is throwing null pointer exception.

And then here you are having cannot add null product. So this will be printed. right. So, that we are testing over there right. So, you have to do all the testing.

So, that is being done right. So, then exception will be caught. So, caught exception and then whatever I said that will be printed all right. So, and then you are having one system dot out dot printl and new line and now next test is handle invalid product removal right. So, you are trying this try throw catch.

So, try right so you are printing this attempting to remove a non-existing product right so kath is invoking remove product so here you have phone comma 800 but the problem is where we have If you recall, we have laptop 1000, right? So, we have laptop only, right? And then remove product.

So, under cart, you have remove product, right? So, here you go. So, products.remove product, when it is calling, right? Remove of product, what it is? It is false.

You cannot remove that product because it is not there in the cart. So, not false become true. So, illegal argument exception, product not found in the cart will be printed, right? So, here Test 4, it will be printed, right.

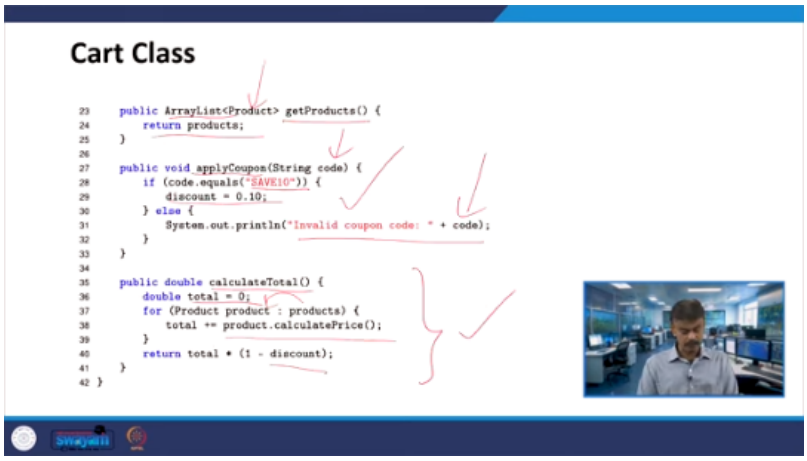
So, that will be caught here, right. And then that particular message will be printed. So, that is test 4. And test 5, applied valid and invalid coupon codes. System.out.println applying valid coupon.

So, caught apply coupon save 10. So, this in fact we had seen that, right. So, if it is save 10, right. So, you can see in the cart save 10 code equals save 10 discount will be 0.10 right. So, this will happen right.

So, that is what you are passing save 10 right. So, then the calculate total that particular function will be called under cart right and then all these messages will be printed this is not very difficult right because you are getting lot of system dot

outdoor print talent. So, this will be printed. And applied coupon. Here it is an invalid code.

Right. So if it is an invalid code. So that also we had seen. Right. Else.

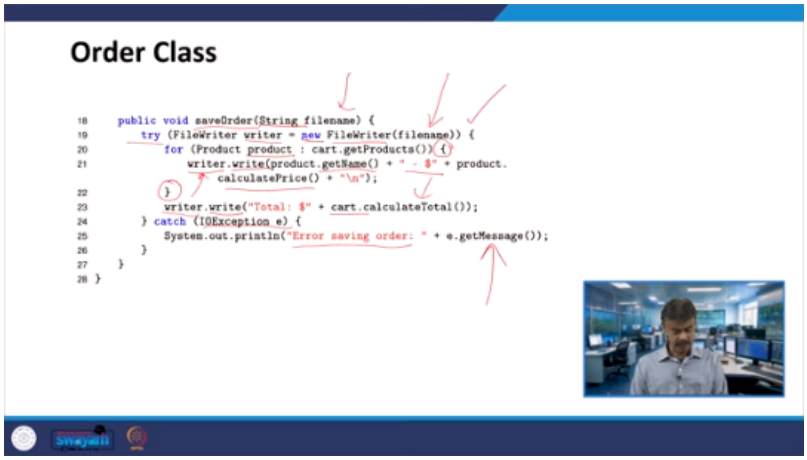


Right. So invalid coupon code. So the code is this. What is that? Invalid code.

Okay. So, System.out.println. Test 5. Which is not very difficult. And test 6.

Confirm correct order summary. Printing this. And then you have an object order. Under class. Right-hand side, you have a constructor.

Passing cart. And order is invoking place order. Right, order is invoking place order. Save order, place order. Okay, so this function will be called. This method will be called, right? Then you are saving the file. So, System.out.println: saving order to file. Order is invoking save order. Order.txt. So, this we had seen in the case of file handling. Last one we had seen, right? So, here you go. Save order file name. What is the file name? Order.txt, right? And then, order saved successfully. So, in case of any problem, right.



So, here because here you are having all the cases right because you are trying right. So, here you are having try yes it is continuing and then finally, catch. In case any exception, it will be caught. All right. So you are all the tests and save the order to a file will be over.

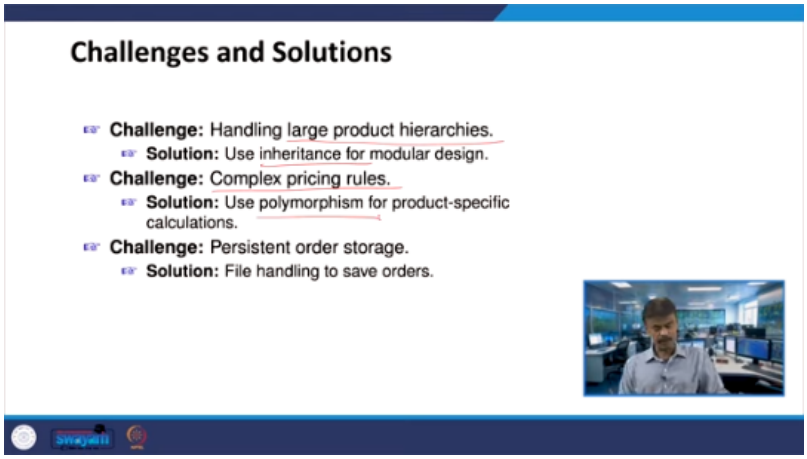
And in fact, when I run the code, I will get the output like this. So what you can do in this case, I will run. Okay. So I'll go to Java. So here you have, you can see.

here you have cot dot class cot dot right cot dot java clothing dot java electronics dot java order dot java right so all the separate right the classes right and also subclasses clothing electronics are subclasses correct so all these you can see and when i run the code so we know the driver class all right so the driver class is nothing but shopping cart demo all right so this is the output right so the different classes i explained right so we had seen card we had seen clothing we had seen electronics right order right and then here you can see order dot txt file also right order dot txt file also correct. So, if I open that order dot txt previous yeah these are all the cases right laptop t shirt and total yeah. So, if you look at the console we have the output like this right.

So, this is what exactly we also expected right. So, this is the output in the console and also we had seen different java file. and output also we had seen so in this case study what are all the challenges and solutions right so in fact the challenge will be handling large product hierarchies right if it is a big supermarket right so you have several products correct and you have to make it online all right so in that case suppose the products are large or heavy So, then when you want to handle, right, so you have to follow few hierarchies, right. So, in fact, we have seen clothing, electronics.

So, like that you have to, I mean, make the heavier one, right. So, the solution will be, of course, we have to use the inheritance, right. So, the inheritance for modular design. And the challenge is complex pricing rule, right? So this is the challenge.

So how to make it sometimes 10% discount, 20% discount and some of the products we have to put GST, right? So all these cases, we have to worry about that, right? So in this case, what we have to do, you can use the polymorphism, right? For product specific calculations and challenge, persistent order storage, right? So in the case of a persistent order storage,



So, the solution is we have to use the file handling to save orders. In fact, in this particular problem, we have done this. So, what are all the further enhancement we can do, right? So, we can add support for additional product categories, right? For example, furniture, grocery items.

So, these are all we have not seen, right? So, you can sports items, right? So, these are all the items we can add, right? And integrate payment gateways for real world use. So, that is the

challenging task nowadays right so we have to integrate the payment gateways and we have to also enhance file handling system for the javascript object notation json and csv right the csv files that means the comma separated values file right for storage and also the last chapter that we had studied the graphical user interface right so we can also add GUI for better visibility. So, these are all the enhancements that we can think of for solving these kind of real world problems. So, in this study right in this case study I talked about the online

shopping cart system right. So, here we built a modular system for simulating an online shopping cart right.

So, we have used several functionalities, and then we had implemented that, right. So, what are all those we have implemented? So, with the help of inheritance, we have done product hierarchies, and with the help of polymorphism, we managed court management and dynamic pricing, right? Also, we have used the file handling system for summary generation and persistent storage, right. So, and also, we ensured robust error handling and exceptions, right?

If there are invalid operations, we have also used error handling exceptions, right? So, with this, I am concluding this particular case study. So, we have studied this with the help of Java, right? Thank you.